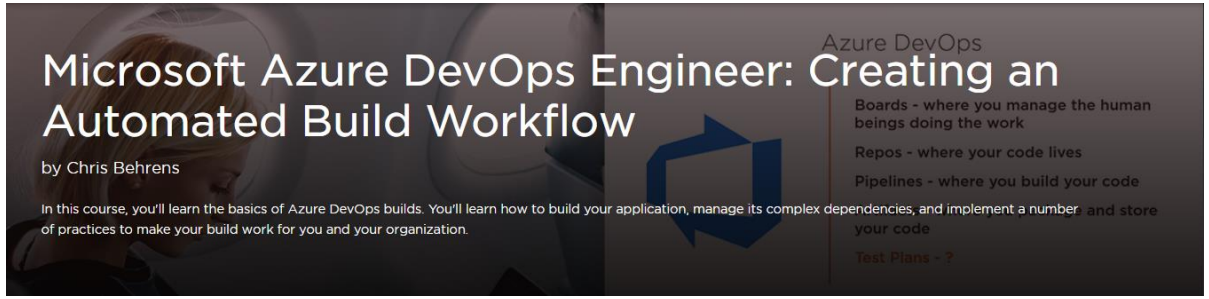
Microsoft DevOps Solutions: Implementing, Maintaining and Standardizing Build Strategies

STRUCTURING YOUR BUILDS

Chris B. Behrens
SOFTWARE ARCHITECT
@chrisbbehrens

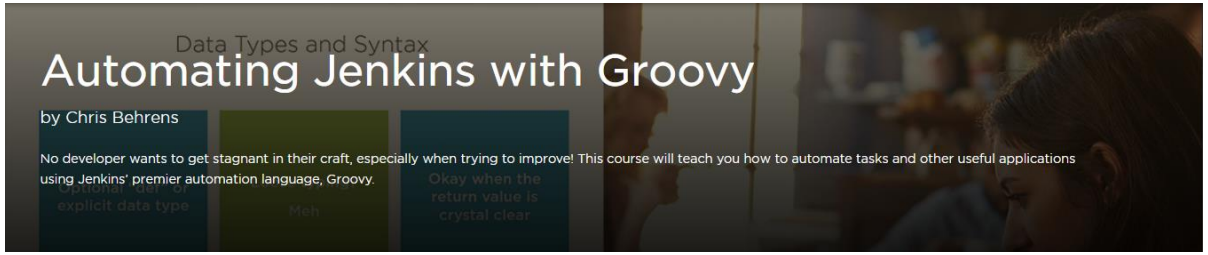
A More Focused Course



https://app.pluralsight.com/library/courses/**microsoft-azure-creating-automated-build-workflow**

https://app.pluralsight.com/library/courses/**microsoft-azure-monitoring-code-quality**

https://app.pluralsight.com/library/courses/**running-jenkins-docker**



https://app.pluralsight.com/library/courses/**automating-jenkins-groovy**

Planning Your Agents

A DotNetCore Agent

One with the DotNetCore build elements installed on it

Or the same for any other build technology

Meta-dependencies

Dependencies can imply other dependencies

Like an OS-specific SDK

Or compiling for a particular processing architecture

Cloud vs. Self-hosted Agents

You may have a tricky build with funky dependencies

There are two models for agents in Azure

An Azure-hosted cloud agent

A You-hosted on-premises agent

Self-hosted Agents

You begin by triggering a build in Azure (more on this in a minute)

Azure reaches out to your network and executes the build there

Via an agent which you install on your target server

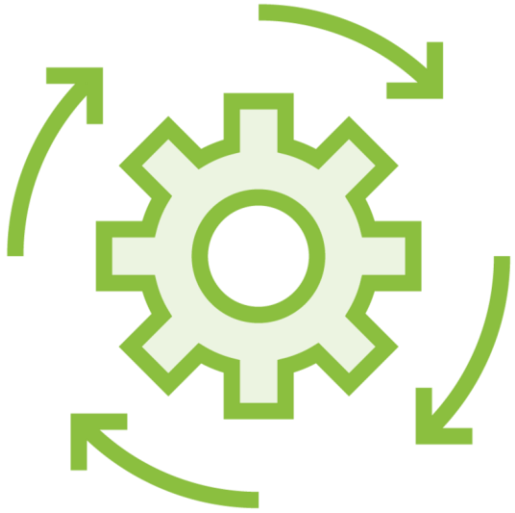
1. Custom dependencies
2. Decreased build latency (Increased Velocity)
3. Incremental Builds

Avoid incremental builds

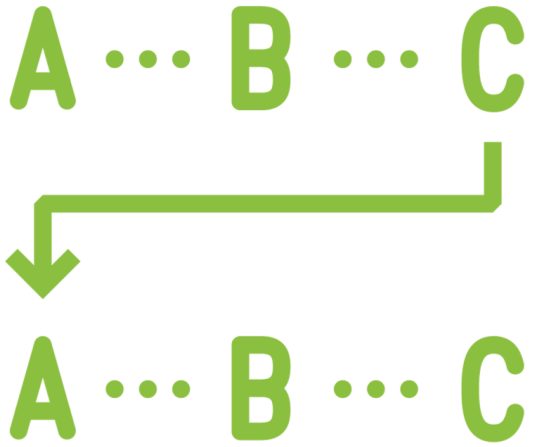
Sometimes you have to live with them

Queued Builds

Soon after you have multiple devs on a project...



You have multiple builds per day...



And soon, you have simultaneous builds queued

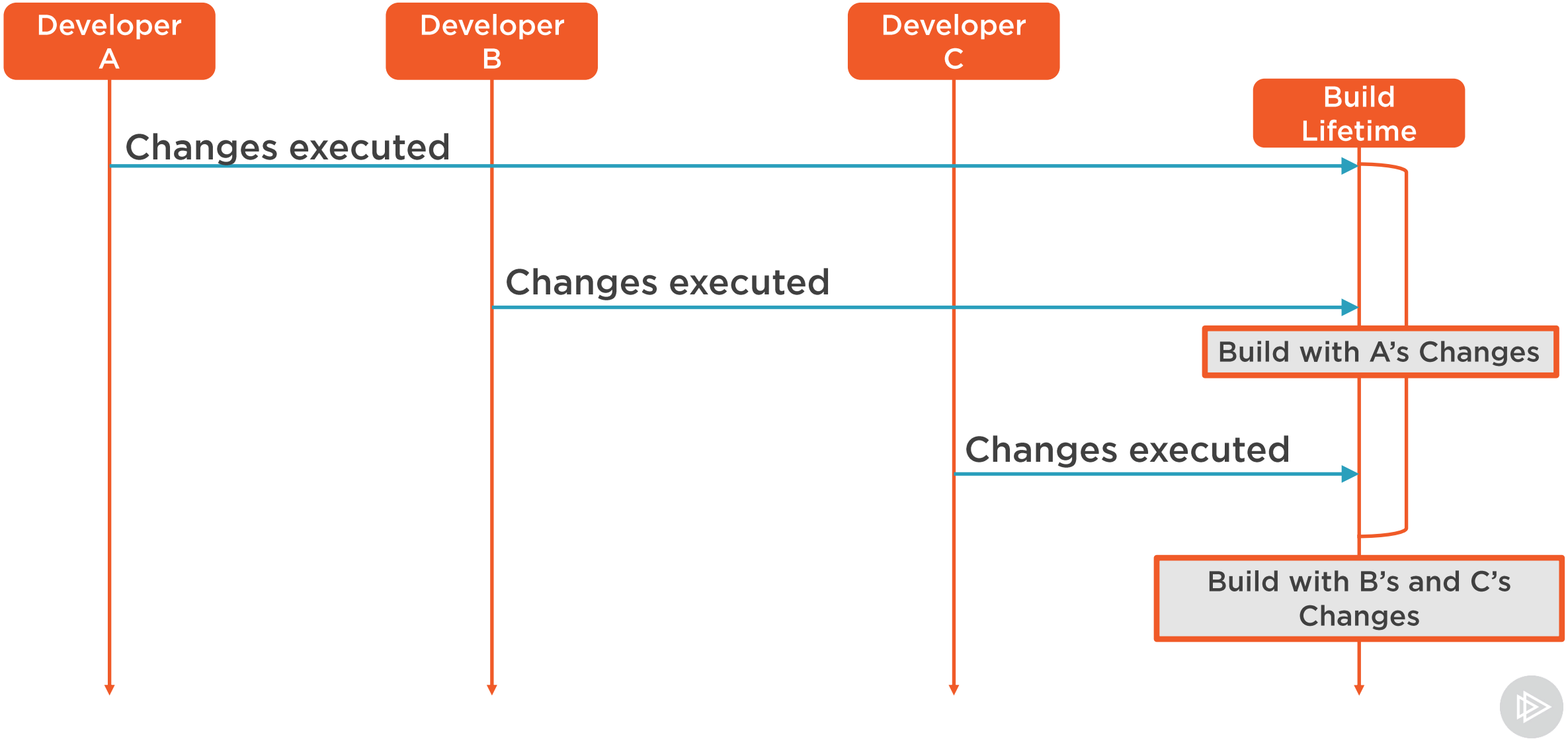
Why Queued Builds are Bad

A developer is waiting

If the two builds' changes are unrelated, they could have been run together

Long-running builds tend to be brittle

Build Batching



Batching and Branching

Batching is limited by version control

If your changes are isolated from each other...

They can't be batched

Sometimes you just need more than one build at a time

Running > 1 agent simultaneously

Microsoft restricts this along these lines:

1. Project visibility
   (public or private)

2. **The number of jobs**

3. **The duration of the job**

Project Visibility	Job Count	Duration
Public	10 free Microsoft-hosted parallel jobs that can run for up to 360 minutes (6 hours) each time	No overall time limit per month
Private	One free job that can run for up to 60 minutes each time	1,800 minutes (30 hours) per month

Project Visibility	Job Count	Duration
Public	Unlimited	None
Private	One self-hosted job; For each active Visual Studio Enterprise subscriber who is a member of your organization, you get one additional self-hosted parallel job.	None

- You’re going to run out of jobs
- But you can simply purchase more
- I’ll show you where and how

Hosting Model	Job Count	Time Limit
Cloud	One free job that can run for up to 60 minutes each time	1,800 minutes (30 hours) per month
Self	One self-hosted job; For each active Visual Studio Enterprise subscriber who is a member of your organization, you get one additional self-hosted parallel job.	None

Creating a Simple Build

master ▾ DNOurWebApp / test-build.yaml

```yaml
# ASP.NET Core
# Build and test ASP.NET Core projects targeting .NET Core.
# Add steps that run tests, create a NuGet package, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-co

schedules:
- cron: "*/5 * * * *"
  branches:
    include:
    - master
    - test-build-branch

pool:
  vmImage: 'ubuntu-latest'

variables:
  buildConfiguration: 'Release'

steps:
- script: dotnet build --configuration $(buildConfiguration)
  displayName: 'dotnet build $(buildConfiguration)'
```

Pipelines are defined in YAML

- A lightweight syntax for properties and values

You define build steps in YAML

And this resides in version control

Right next to your code

1. **Developer creates a feature branch**
2. **Developer works on a feature branch, and commits the changes**
3. **Developer creates a Pull Request**
4. **Azure DevOps triggers a build**

SonarQube or PMD for static analysis

Open-Source vulnerability scan with WhiteSource Bolt

Penetrating testing with OWASP ZAP

Our Current Build Trigger

test-build.yaml

```
trigger

    -master
```

The build definition flows with the code along the branch

The trigger definition does not – this *master*-based trigger is on the *test-build-branch* branch and so will not be fired

At least, not until we merge it to *master*

Our Current Build Trigger

test-build.yaml

```
trigger
    -master
    -test-build-branch
    batch: true
```

Let’s say we need the build to do our work

Maybe we can’t debug locally and we need the build to create a test environment

Try to avoid this…

Trigger Types

1. Push trigger
2. Pull Request trigger

```
pr:
    -master
    -develop
```

3. Scheduled trigger
4. Pipeline trigger

Our Build Trigger

```
trigger
    master
    test-build-branch


trigger:
    branches:
        include:
            -master
            -test-build-branch
```

Our Build Trigger

```
branches:
        include:
            test-build-*
        exclude:
            test-build-branch
```

Path Exclusions

Trigger on everything except docs/

```
trigger:
    branches:
        include:
            test-build-*
        exclude:
            test-build-branch
    paths:
        exclude:
            docs/
```

Trigger ONLY on docs/

```
trigger:
    branches:
        include:
            test-build-*
        exclude:
            test-build-branch
    paths:
        include:
            docs/
```

Tag-Driven Triggers

```
trigger:
    tags:
        include:
            buildme
        exclude:
            dontbuildme
```

Build-Skipping Commit Messages

- [skip ci] or [ci skip]
- skip-checks: true or skip-checks:true
- [skip azurepipelines] or [azurepipelines skip]
- [skip azpipelines] or [azpipelines skip]
- [skip azp] or [azp skip]
- ***NO_CI***

Orchestrating Multiple Builds in a Pipeline

Build complexity will reflect your operational complexity

With multiple servers for multiple application elements

Some elements will be partially dependent...

But independent enough to need their own build

Interdependent Builds

An advanced build which compiles and validates your code

The application depends on a SQL Server database

The Swiss Army Knife Build (A Bad Thing)

The schema migration build needs to precede the app build

And it needs to be tied to the same branch

It's easy to just load up your build with both parts

Why Builds Should Be Independent

You may need to build that tool for other reasons

Maybe a new developer is setting up their development environment

Single Responsibility Principle

Pipeline Triggers in YAML

```
# this is being defined in app-ci pipeline

resources:

  pipelines:

  - pipeline: securitylib   # Name of the pipeline resource

    source: security-lib-ci # Name of the pipeline referenced by the pipeline
resource

    trigger:

      branches:

      - releases/*

      - master
```

A Simpler Example

The db migration stuff is solid, but complex

Let's dial it back to two simple assets

Building a Docker container

Building our code